WinFS Replication

Doug Terry
Microsoft Research Silicon Valley

June 2003

WinFS Synchronization & Caching Team

Mission: *Provide a state-of–the-art synchronization platform integrated into and leveraging WinFS in support of diverse applications requiring peer-to-peer replication for data sharing, high-availability, and offline access.*

Irena Hudis – Product Unit Manager
Lev Novik – Architect

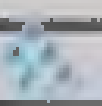
Talk Outline

This morning:

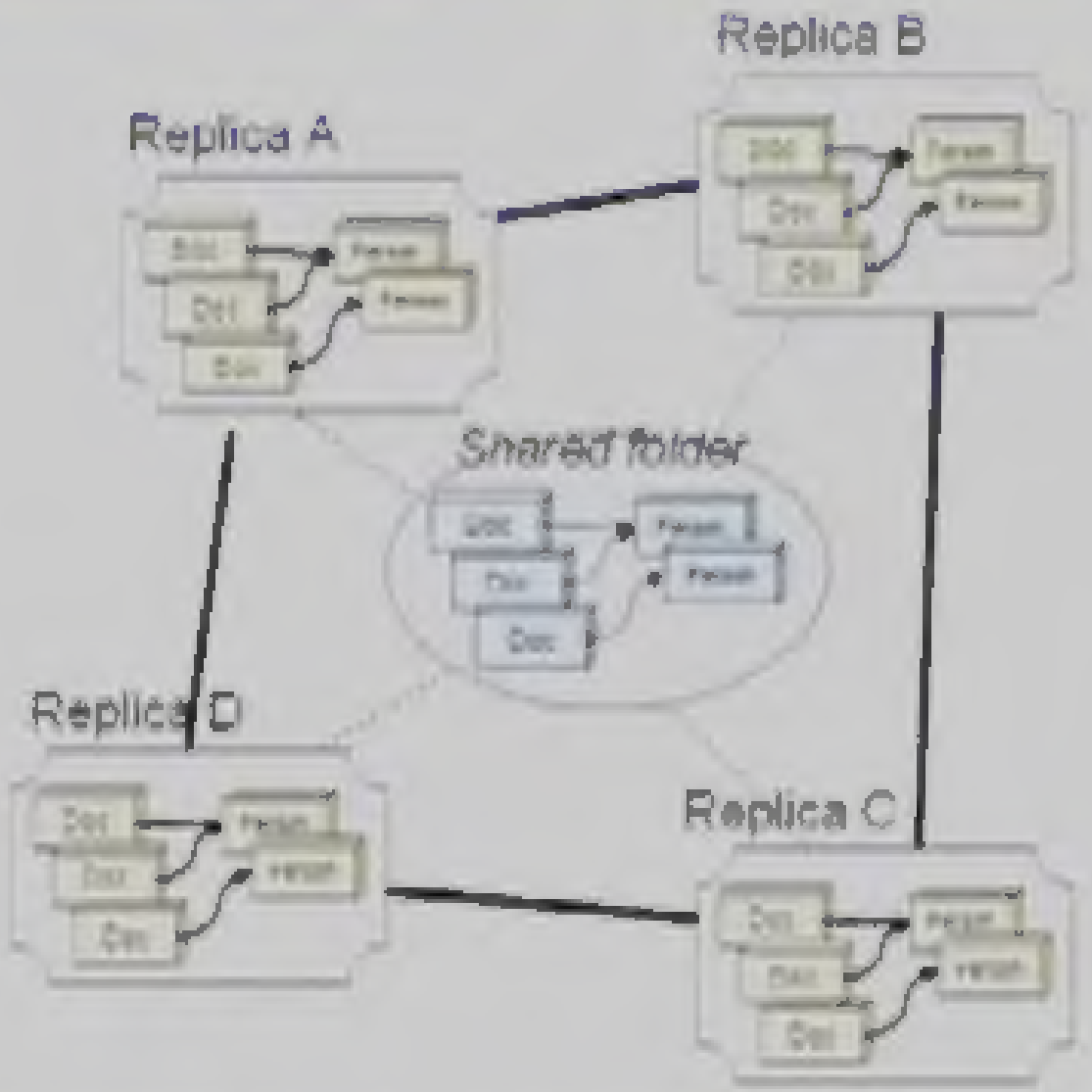
- Goals and requirements
- Change units, versions, knowledge
- Update propagation
- Conflict detection and resolution

This afternoon:

- Filtered sync
- Meta-data cleanup
- Open Issues



Goal: Replication of WinFS Folders



WinFS Synchronization Requirements

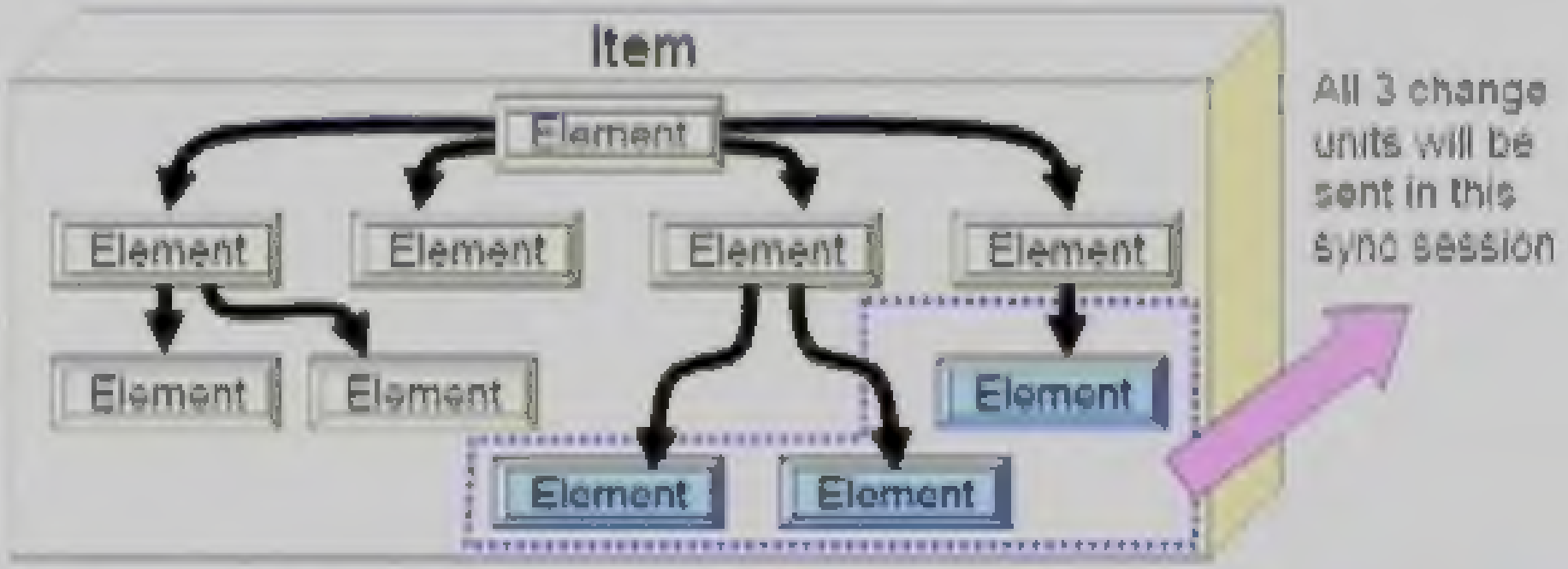
- Allow multi-master updates
- Support peer-to-peer topologies
- Operate at the level of WinFS Schema
- Permit configuration of "what" and "when" to sync
- Detect and handle conflicts during synchronization
- Accommodate varying network characteristics: intermittent connections, slow networks, offline
- Provide out-of-the-box "zero admin" manageability, resilience, and ease of use
- Guarantee convergence, even in "unmanaged" environments

WinFS Synchronization – Approach

- Read-anywhere/update-anywhere model
- Pair-wise reconciliation between replicas
- User-defined topology and schedule
- "Net Changes" of WinFS items/folders are conveyed, not individual actions
- Adapters for replication with non-WinFS machines, services, and devices (*not discussed in this talk*)

Items vs. Change Units

- Item = granularity of consistency
- Change unit = granularity of update, propagation, and conflict detection
  - elements or fields of elements



Versions of Changes

- Each change has a globally-unique id (version)
- Version = ID of the replica making the change + replica-specific number
  - Replica IDs are GUIDs
  - Replica-specific number is ever-increasing at the replica
  - e.g. A5, B2, C7
- Version of change unit is version of last change applied to that element
  - e.g. i.c: A5
- Versions only used by synchronization protocol; not visible to WinFS users

Knowledge for Update Propagation

- Knowledge = concise description of the set of changes received by a replica
  - knowledge depends only on local database
  - not pair-specific: not "what I have received from you"
- When requesting changes, specify
  - Current knowledge (i.e. "*don't send these*")
  - Filter (i.e. "*only interested in these*")
- When enumerating changes
  - Return change units and tombstones not covered by requester's knowledge
- When conveying changes, specify
  - Version of the change
  - The change itself
  - 'Made with' knowledge – for conflict detection
  - 'Learned' knowledge – not to send them again

Knowledge Abstractly

Main operations on knowledge:

- Test if a given knowledge covers a given change
  - "Does he already know about this change?"
- Add a change version to one's knowledge
  - "I just learned of a new change."
- Merge one knowledge with another to produce combined knowledge
  - "I knew this and he knew that, and now we both know this."

Knowledge Implementation

- Version vector = set of (replica GUID, replica-specific number) pairs
- Semantics: "all versions authored by this replica up to this number"
- Exceptions needed for changes received out-of-order
- Includes every replica that ever made a change

Protocol Using Knowledge

Machine A

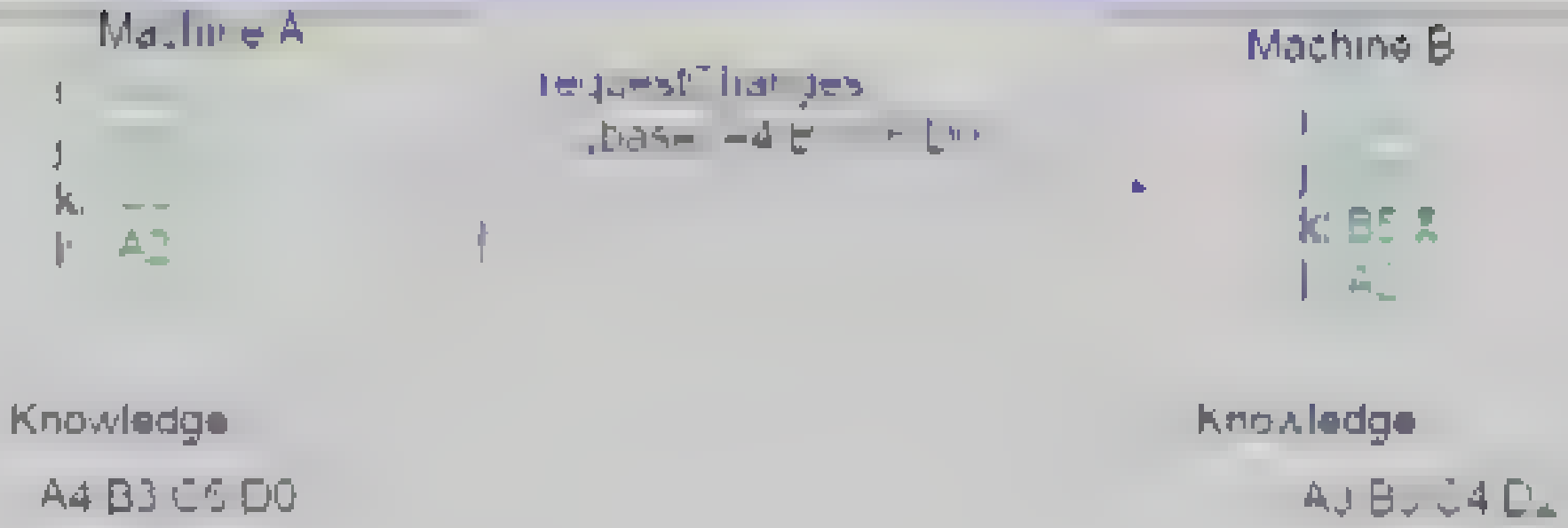
Machine B

Knowledge

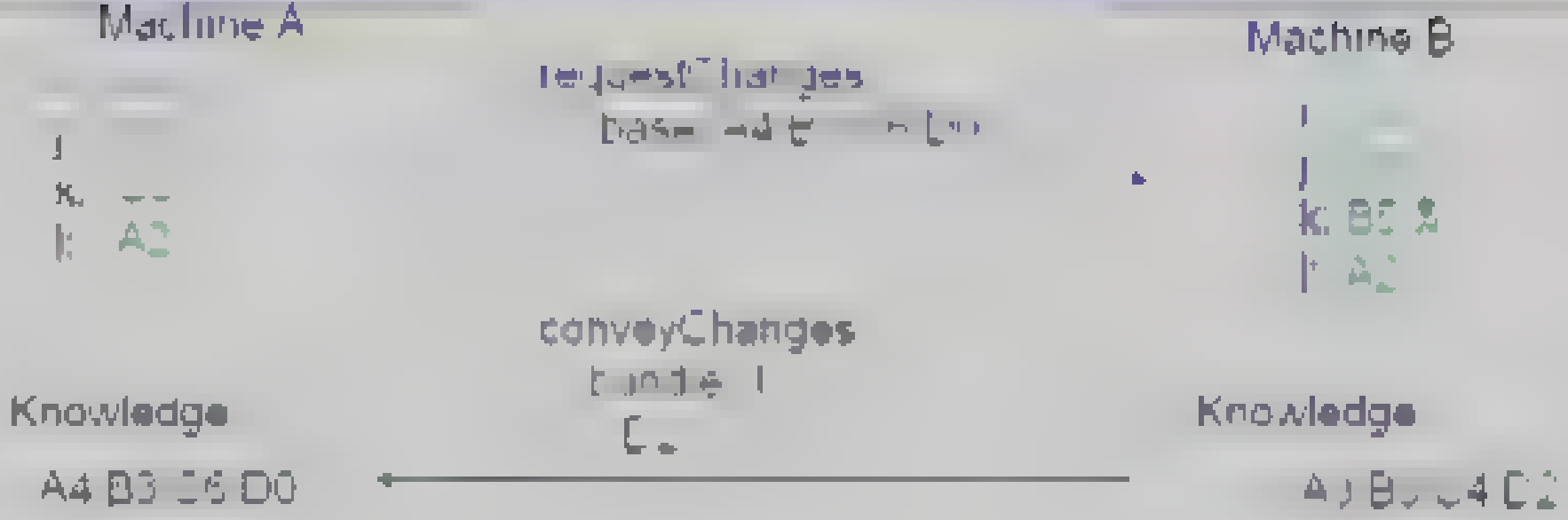
A4 B3 C6 D0

Knowledge

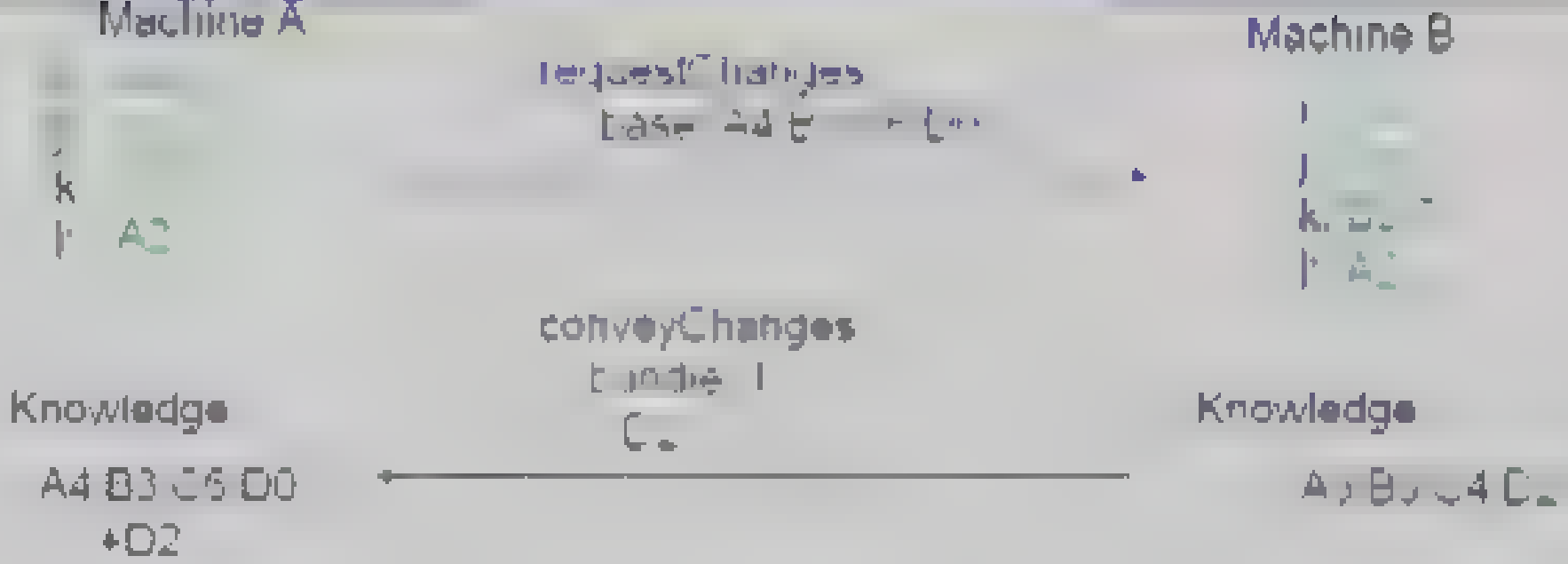
Protocol Using Knowledge



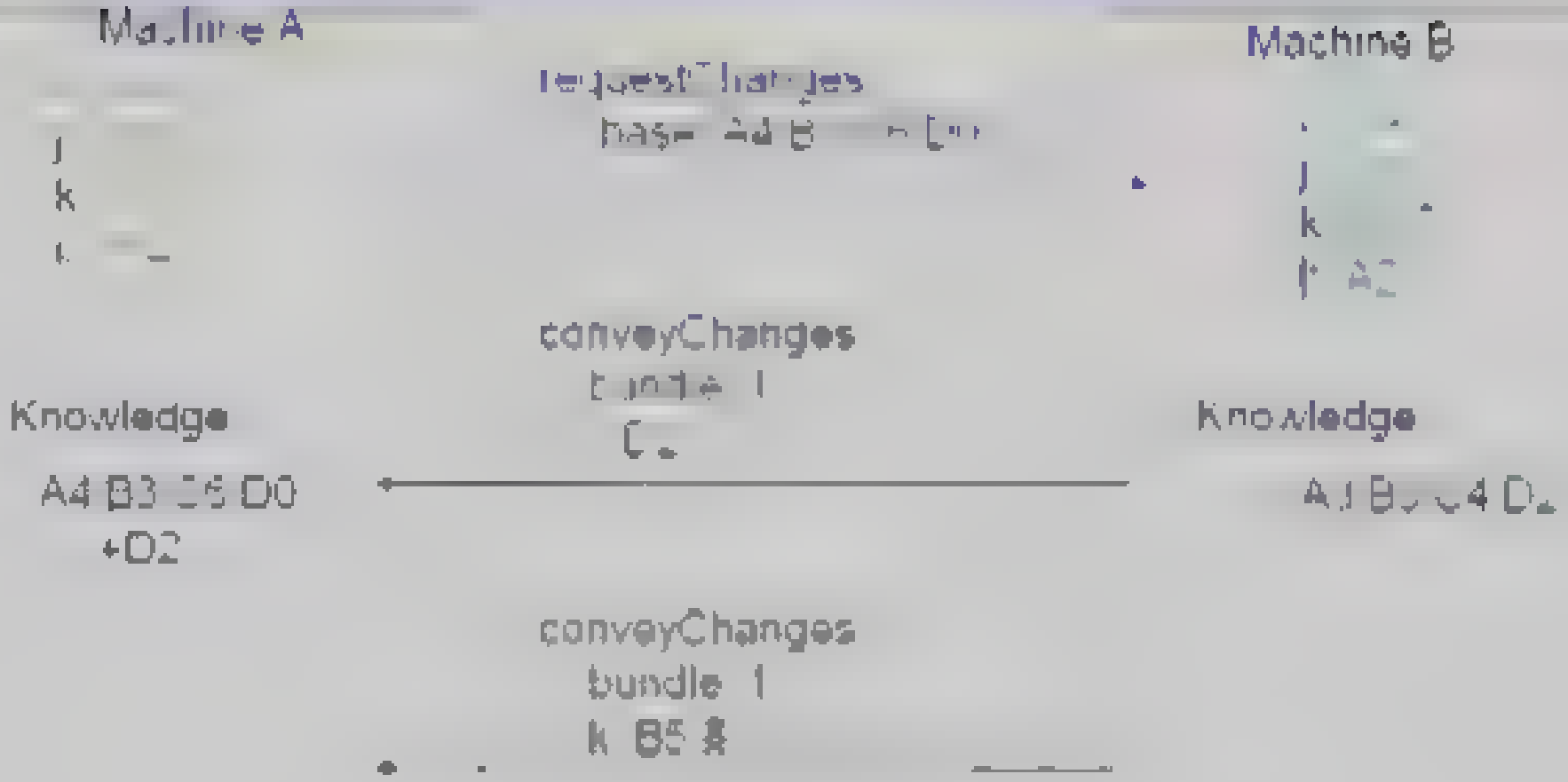
Protocol Using Knowledge



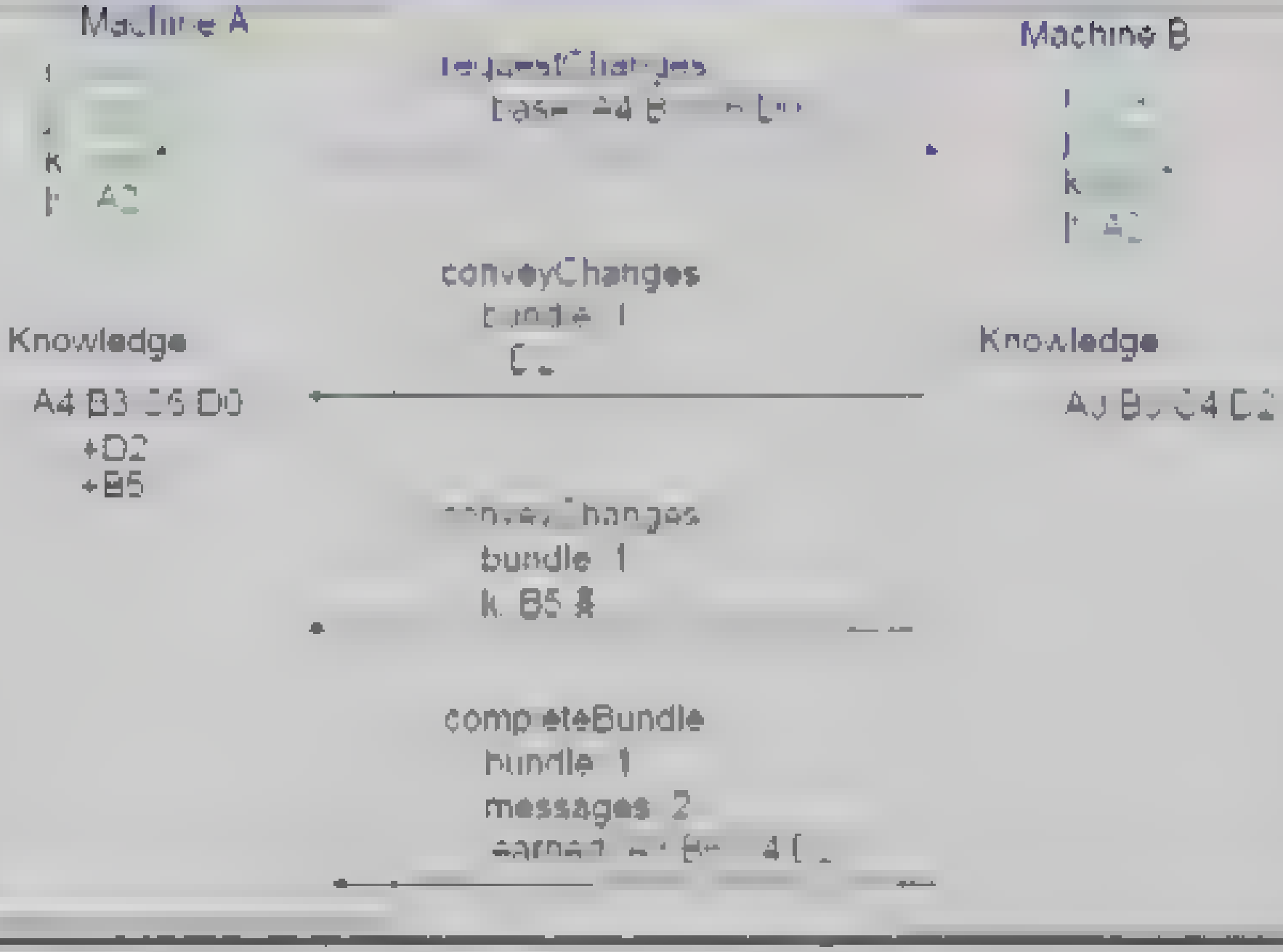
Protocol Using Knowledge



Protocol Using Knowledge



Protocol Using Knowledge



Sync Protocol : Things Not Shown

- Conflict detection and resolution
  - [illegible] concurrent up[illegible]
  - [illegible]ng or automatica[illegible]
- Batching
  - [illegible]
  - [illegible]
- Topology optimizations
  - [illegible]
- Filtering
  - [illegible]
- Meta-data cleanup
  - [illegible]
  - [illegible]

Conflict Definition

- Two changes conflict iff

  (1) the semantics of the changes are incompatible

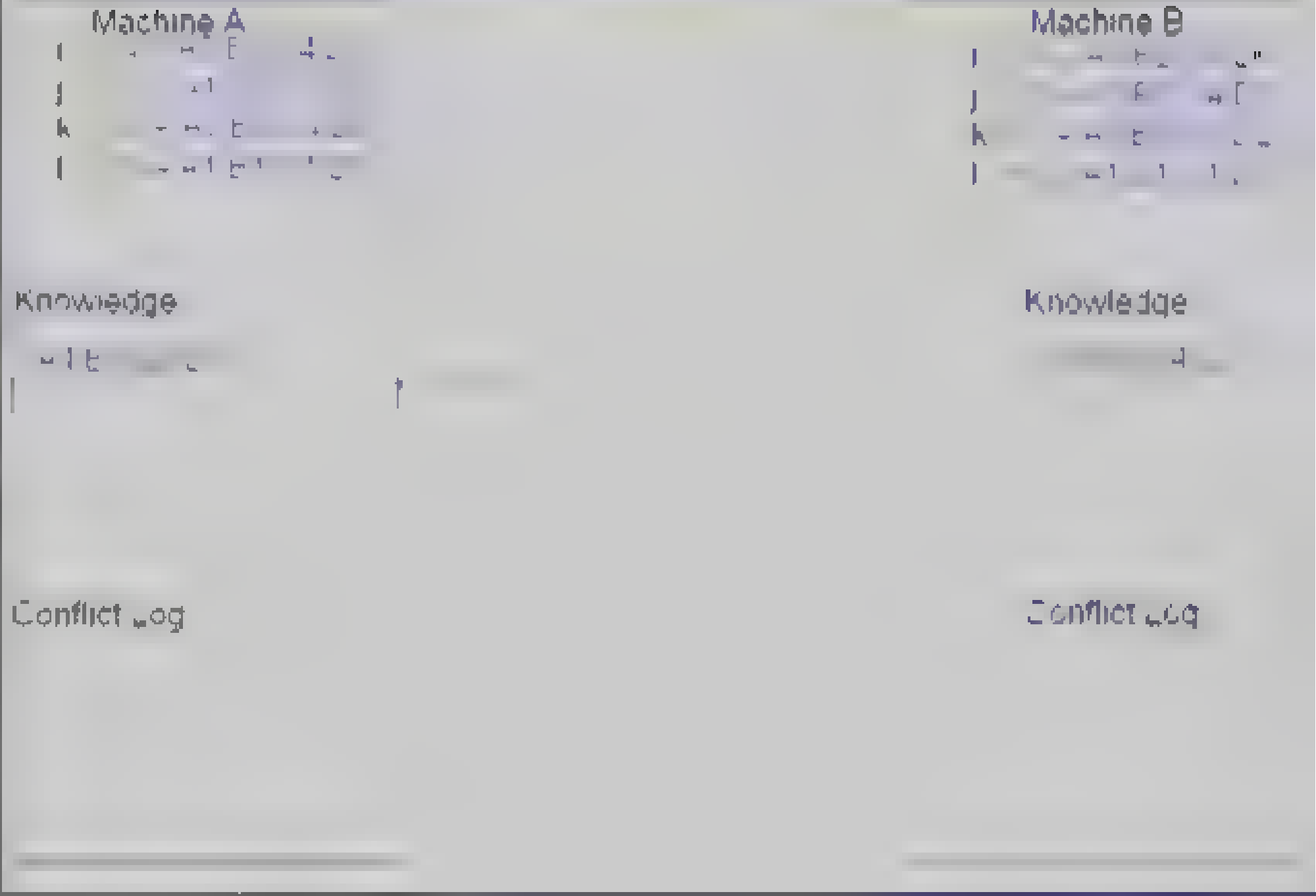
  (2) the changes were performed concurrently
  (without knowledge of each other)

Conflict Detection using Knowledge

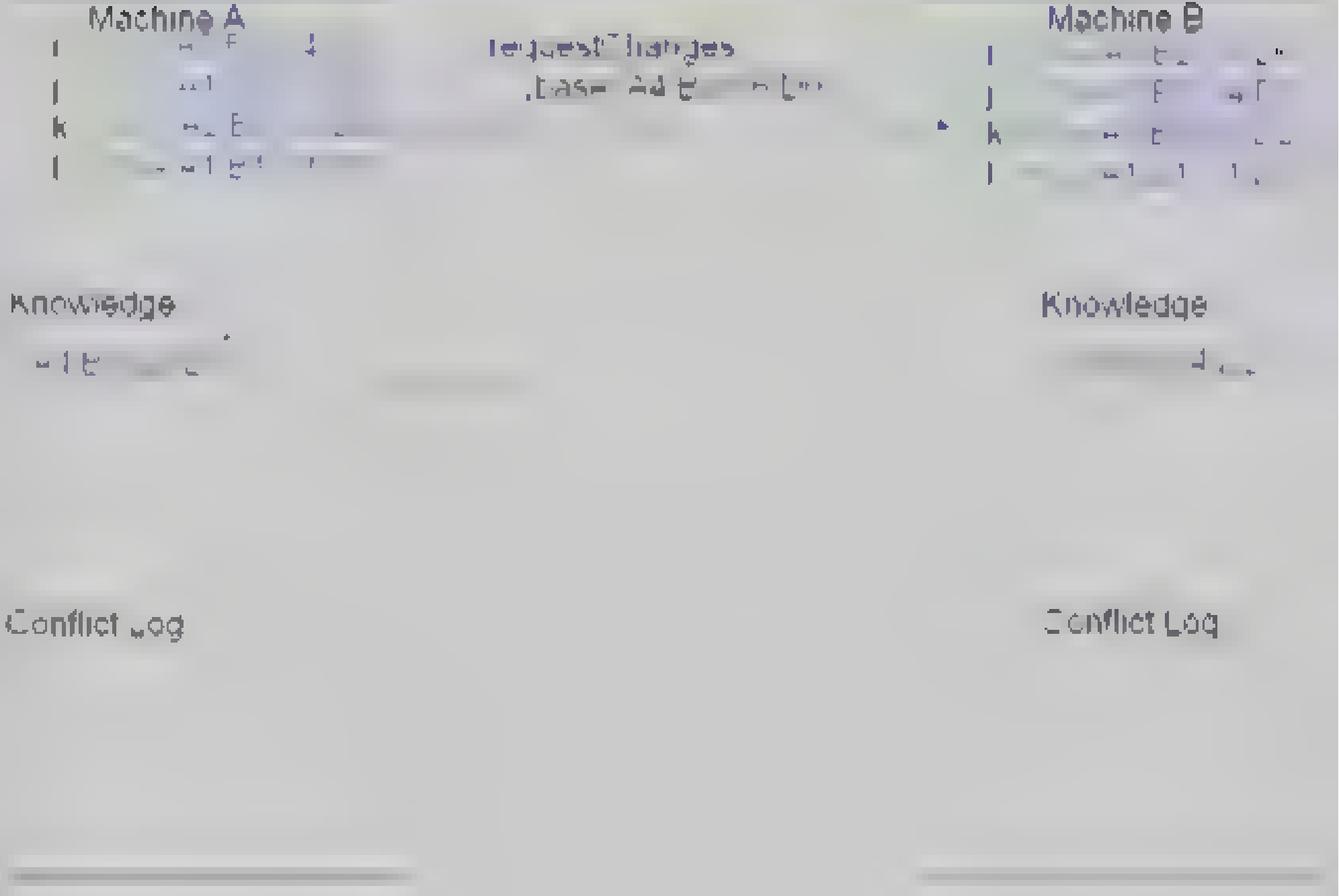
- "Made with" knowledge = knowledge of replica when change was accepted

- Changes C_1 and C_2 are concurrent iff
  (1) C_1 version not in C_2 madeWithKnowledge and
  (2) C_2 version not in C_1 madeWithKnowledge

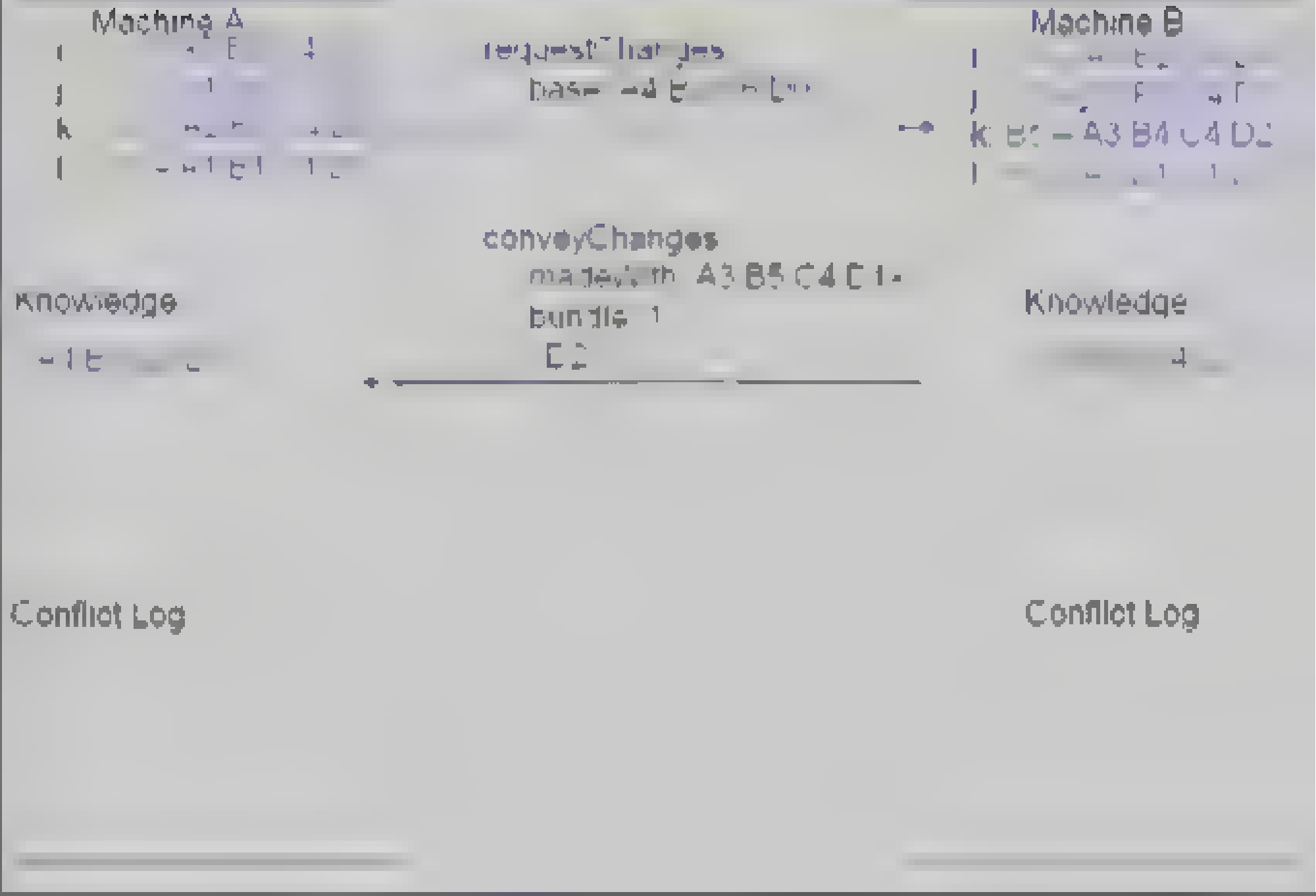
Protocol with Conflict Handling



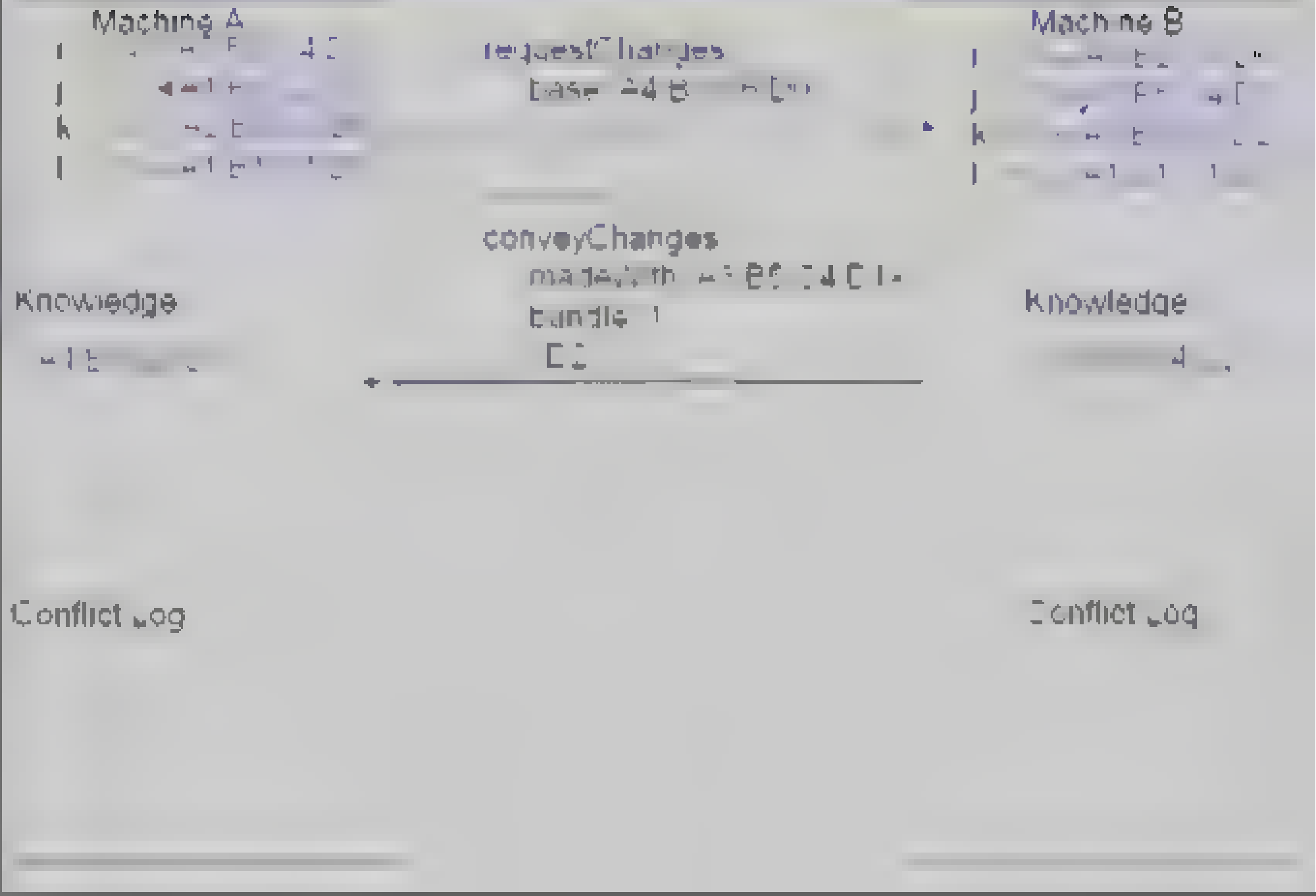
Protocol with Conflict Handling



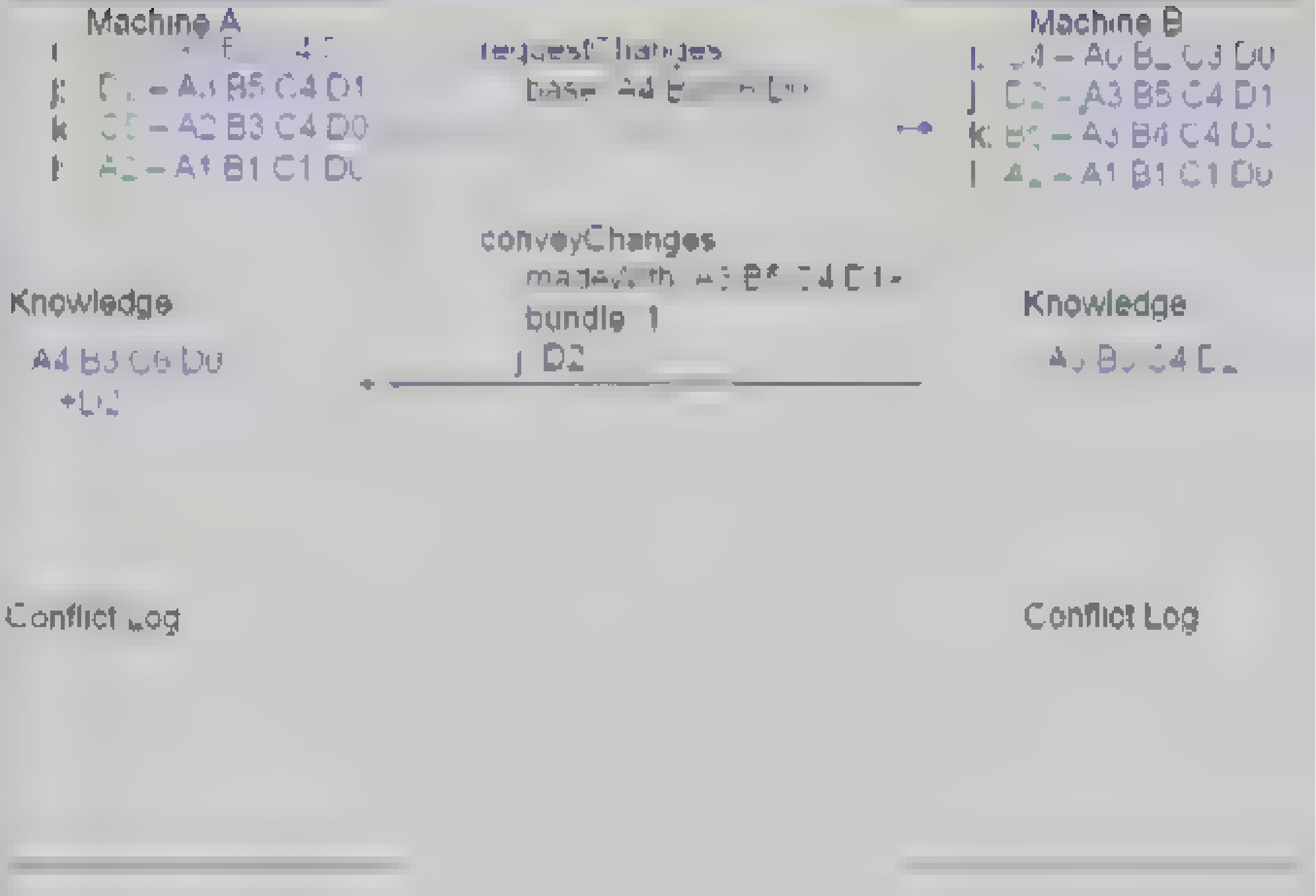
Protocol with Conflict Handling



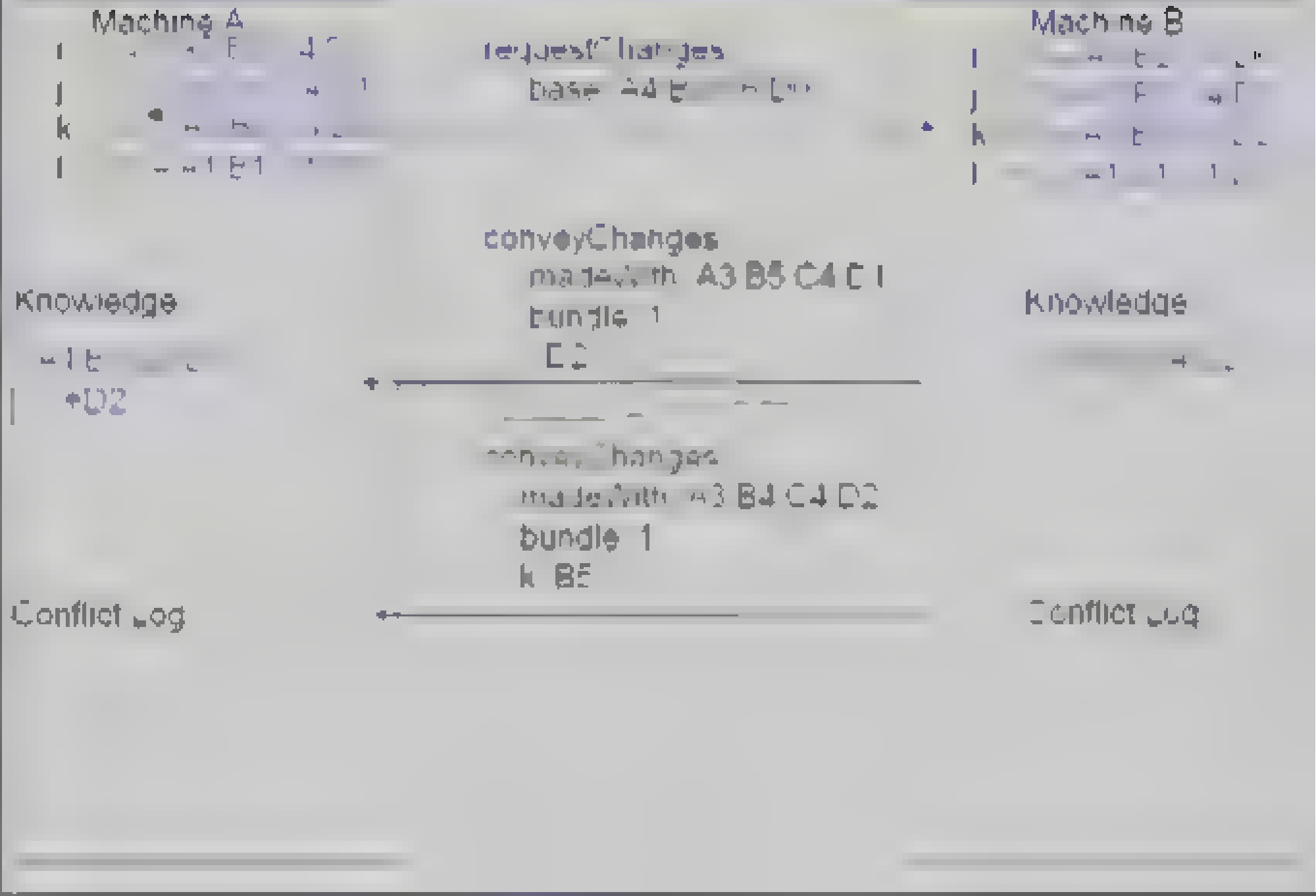
Protocol with Conflict Handling



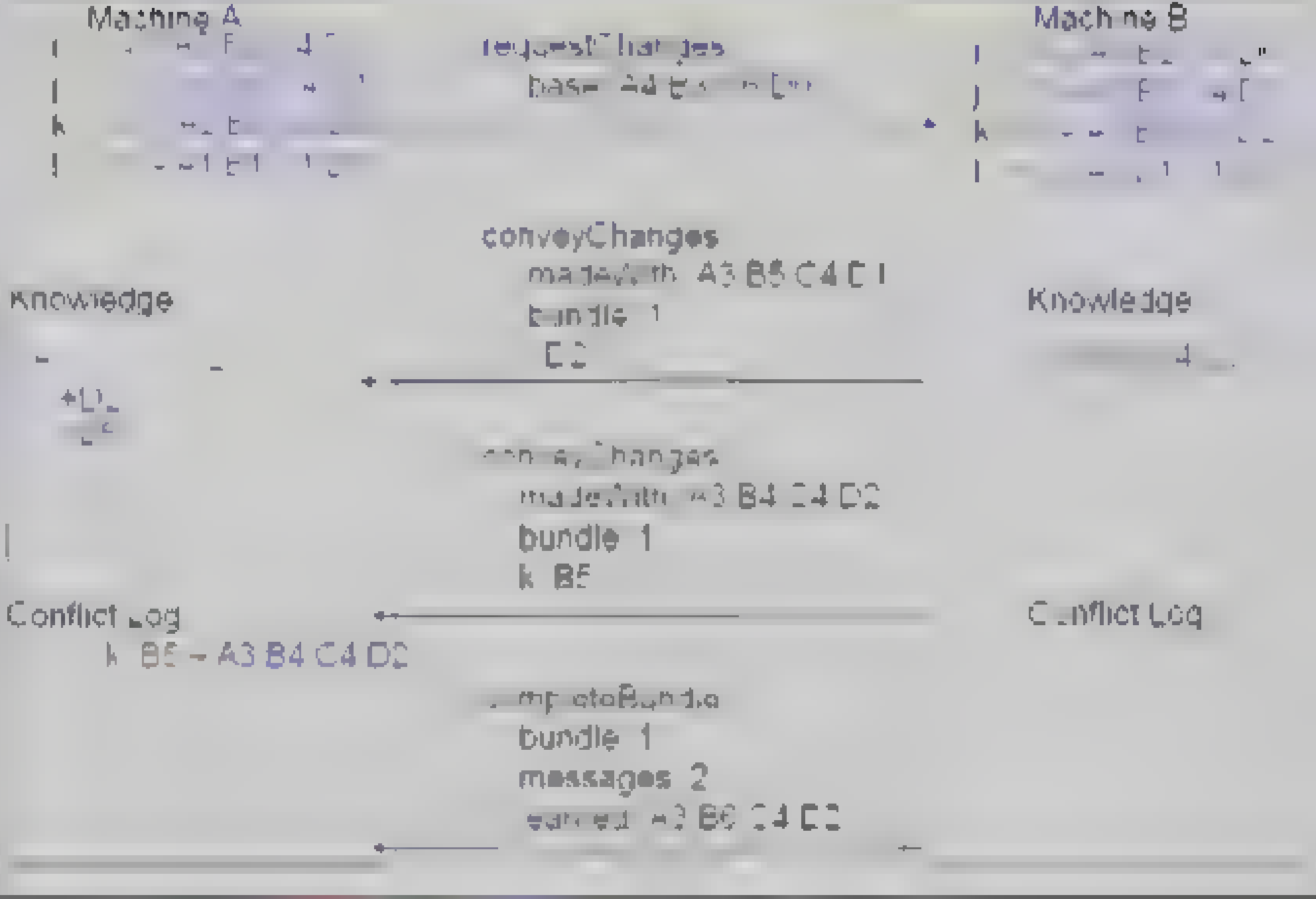
Protocol with Conflict Handling



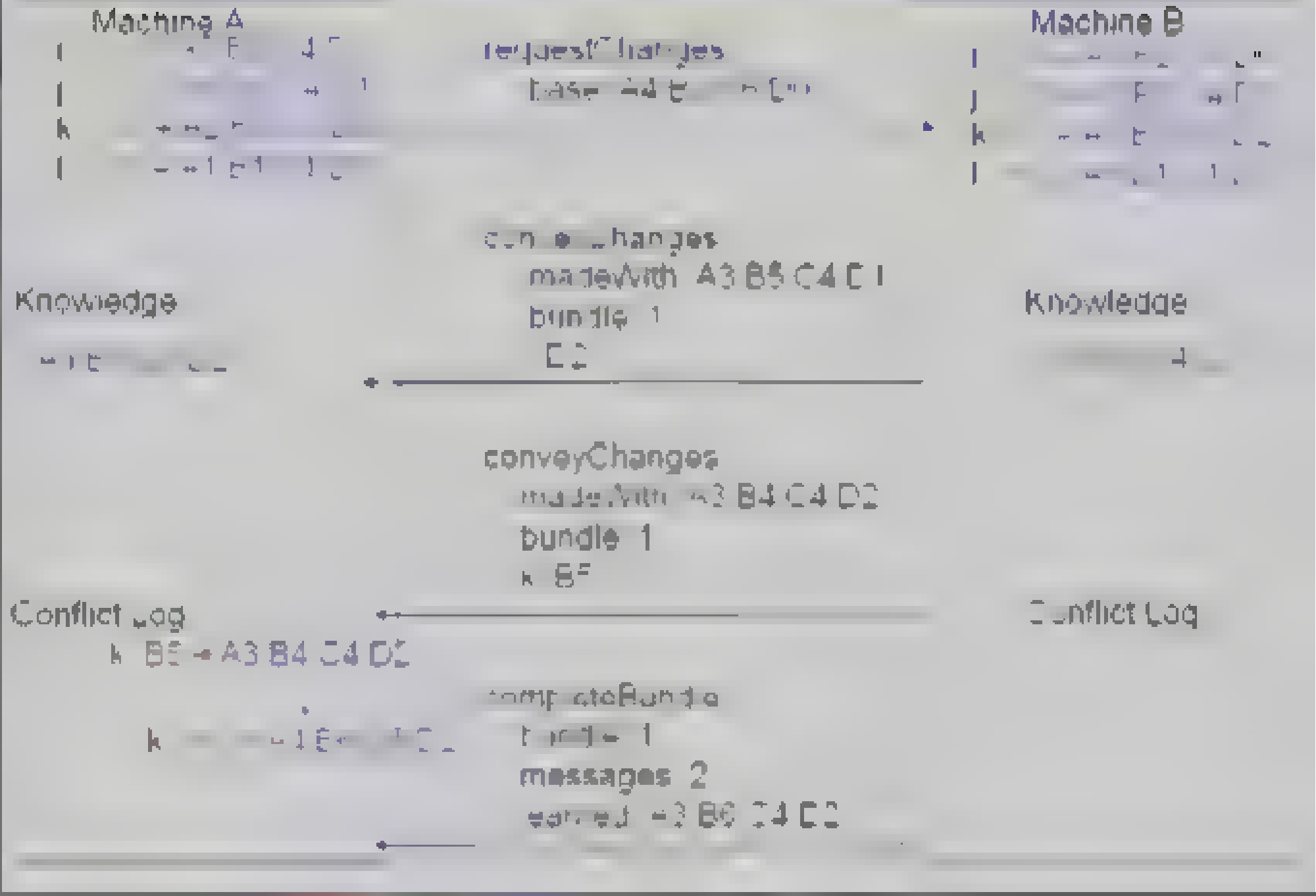
Protocol with Conflict Handling



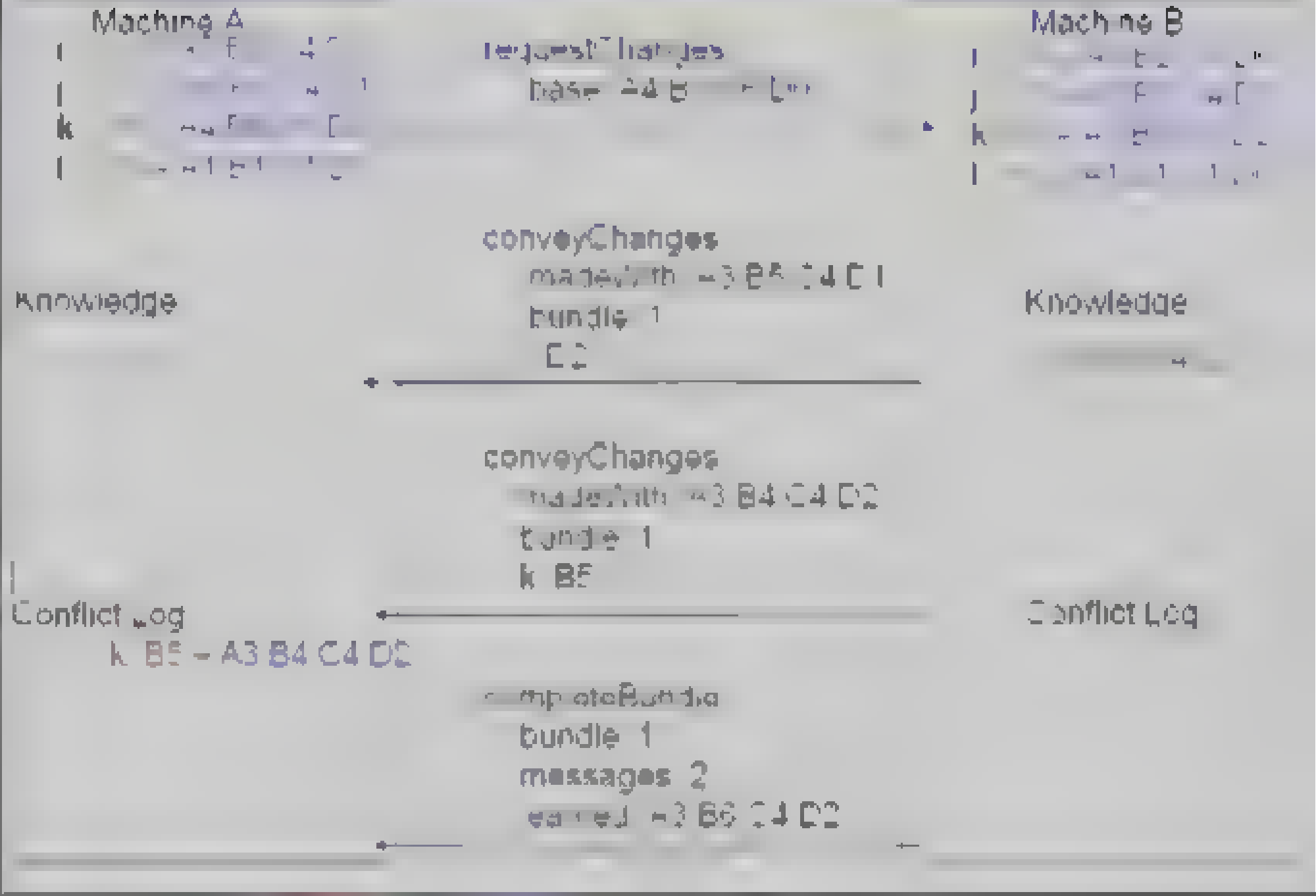
Protocol with Conflict Handling



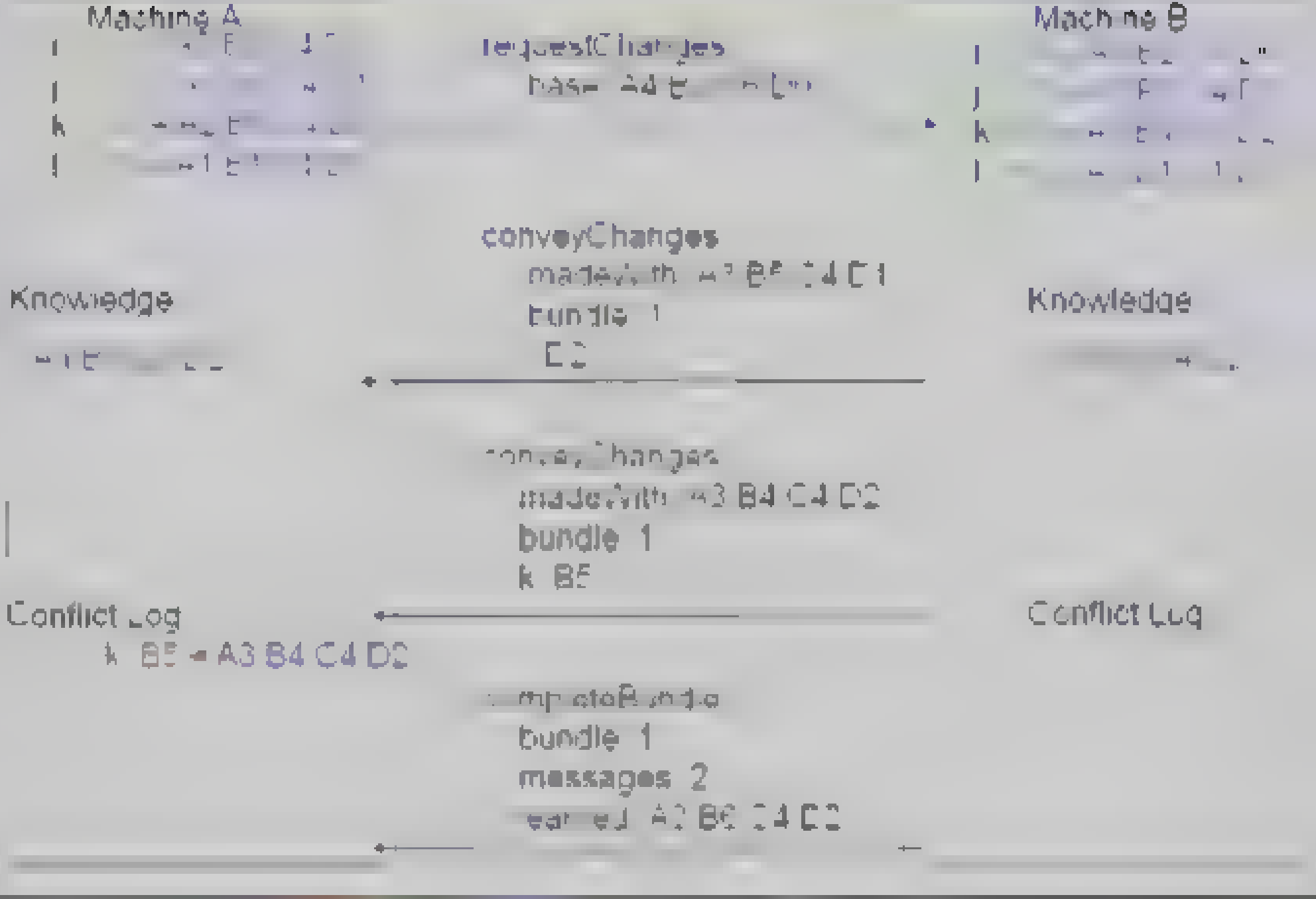
Protocol with Conflict Handling



Protocol with Conflict Handling



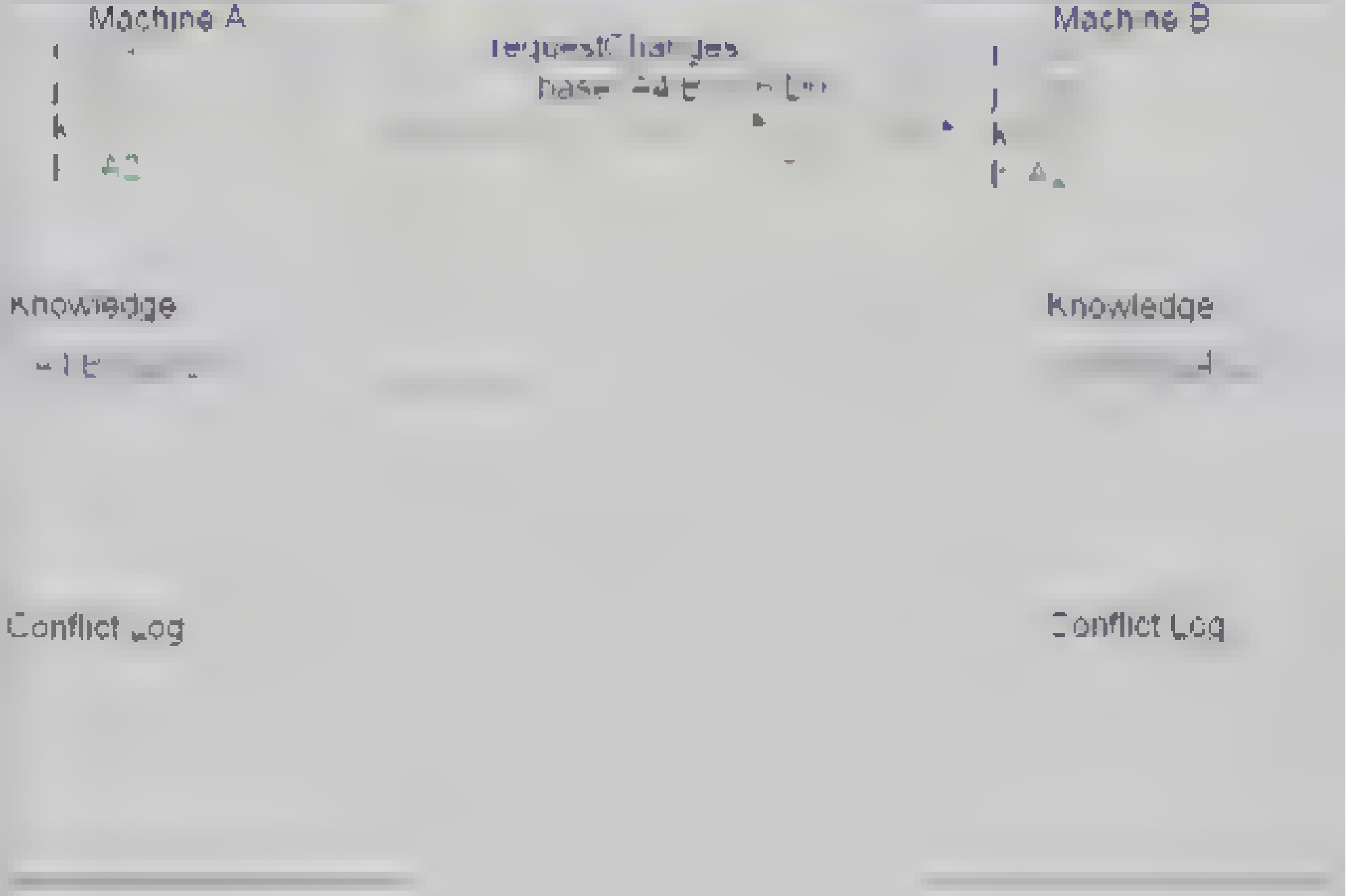
Protocol with Conflict Handling



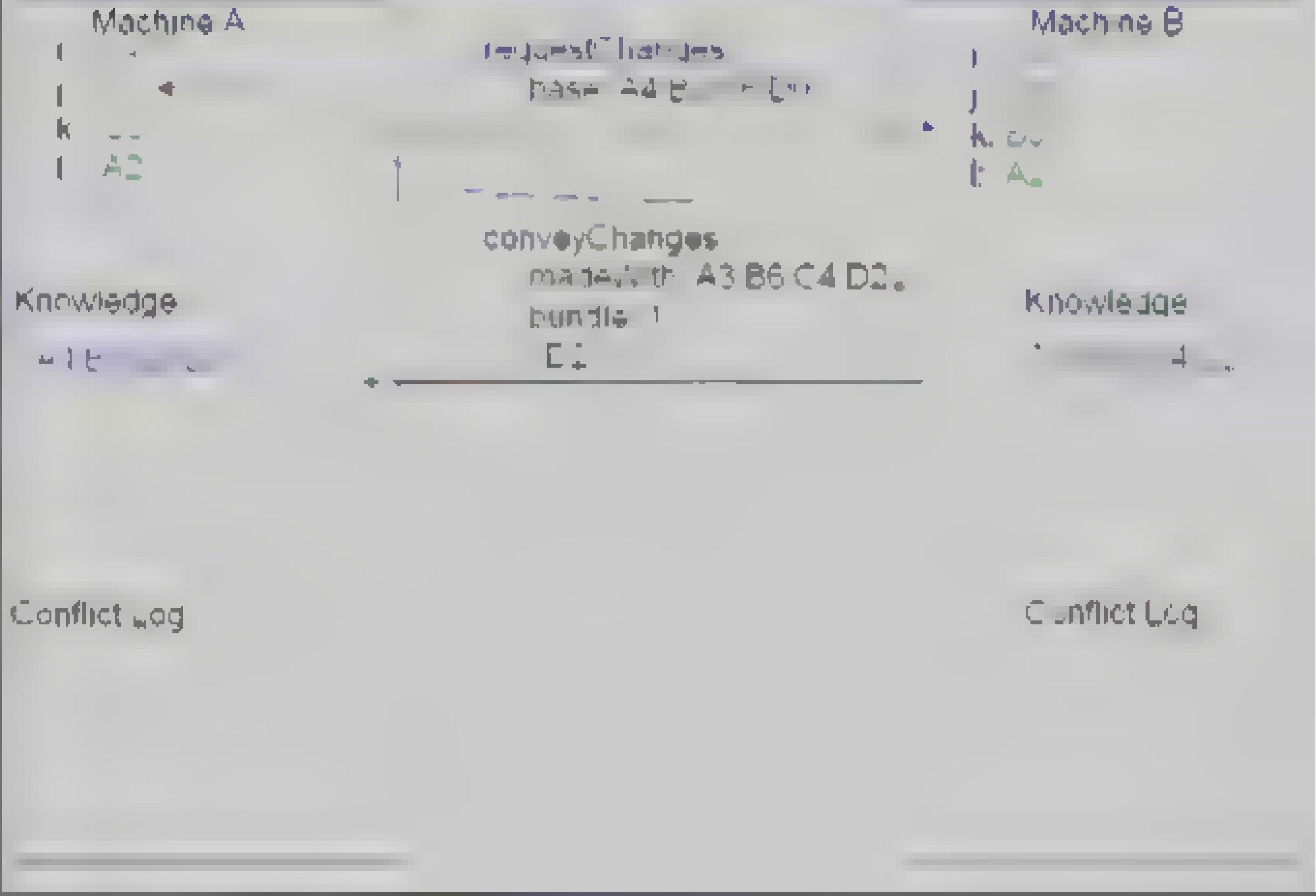
Protocol with Conflicts: Not Shown

- Automatic Conflict Resolution
  - [illegible]
  - [illegible]
  - [illegible]
  - [illegible]
- Storage Optimization
  - [illegible] conflict
  - [illegible]

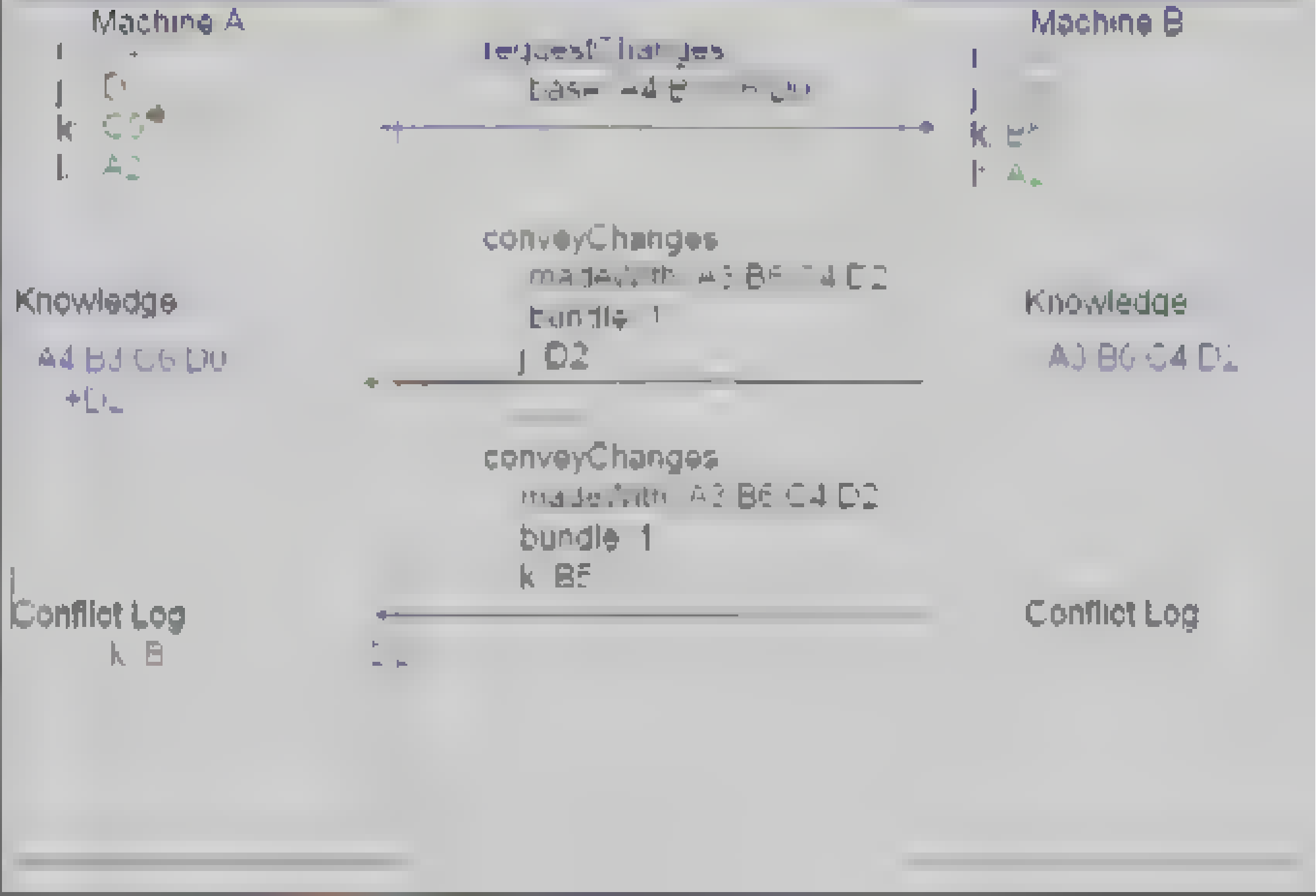
Protocol with Implicit "Made With" Knowledge



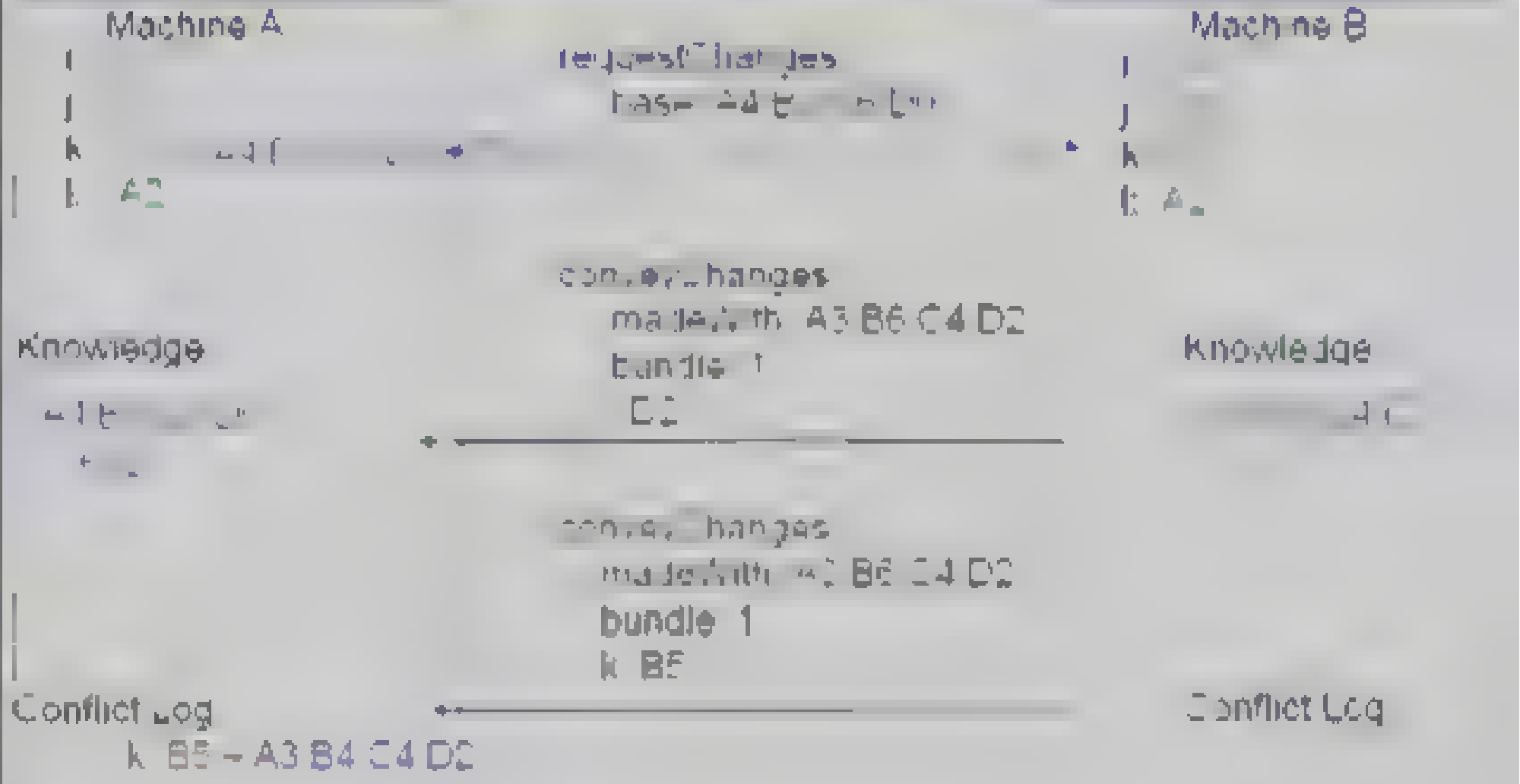
Protocol with Implicit "Made With" Knowledge



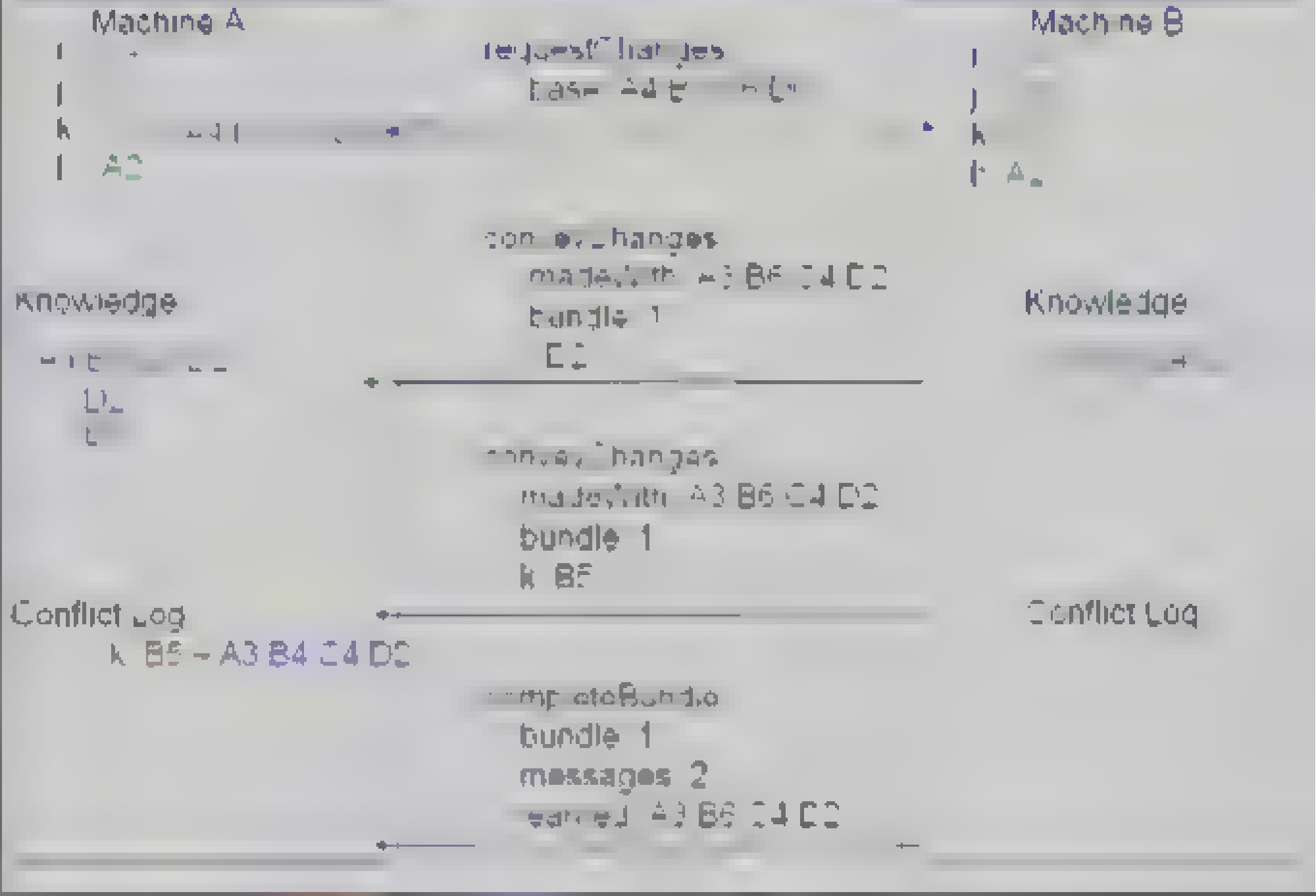
Protocol with Implicit "Made With" Knowledge



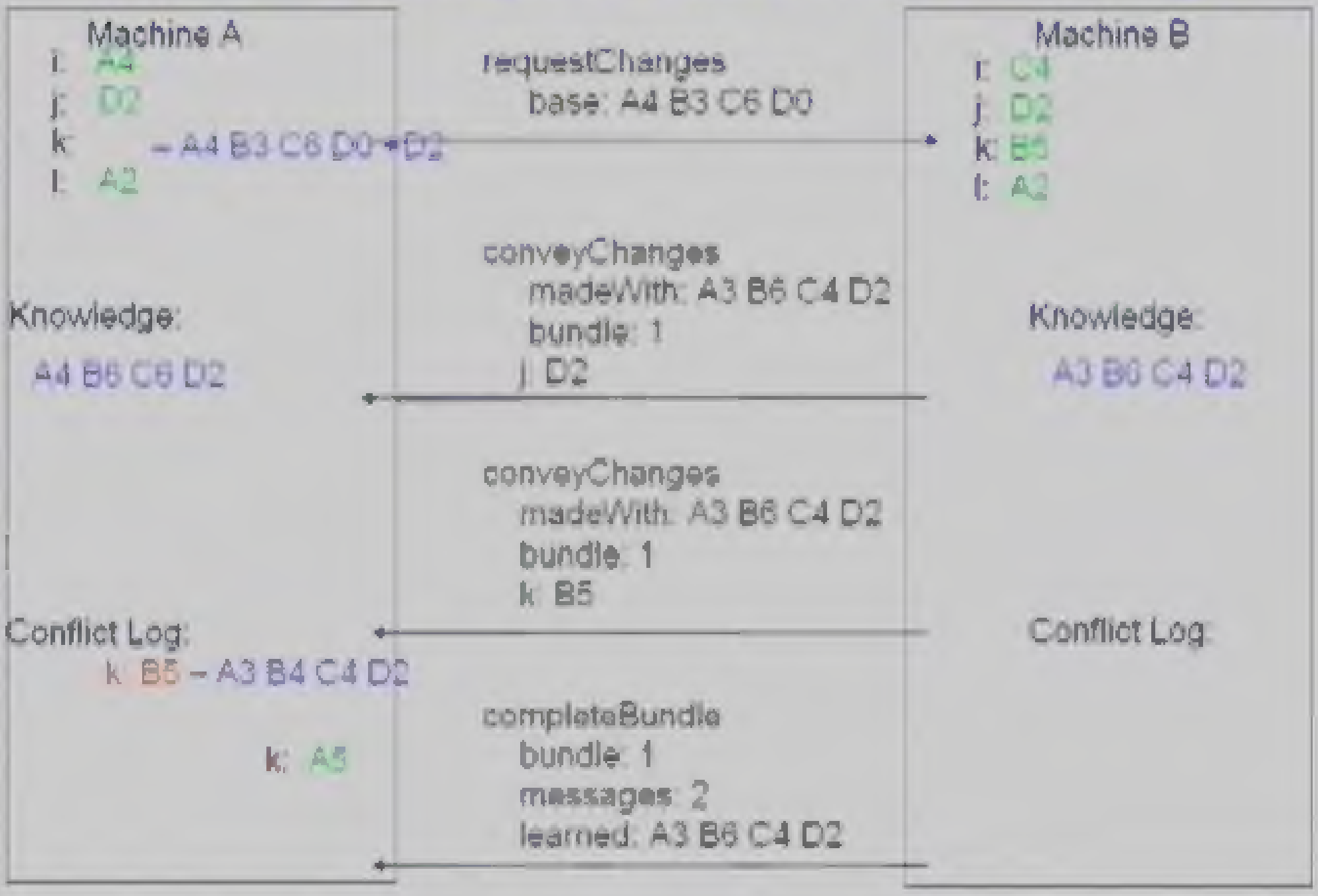
Protocol with Implicit "Made With" Knowledge



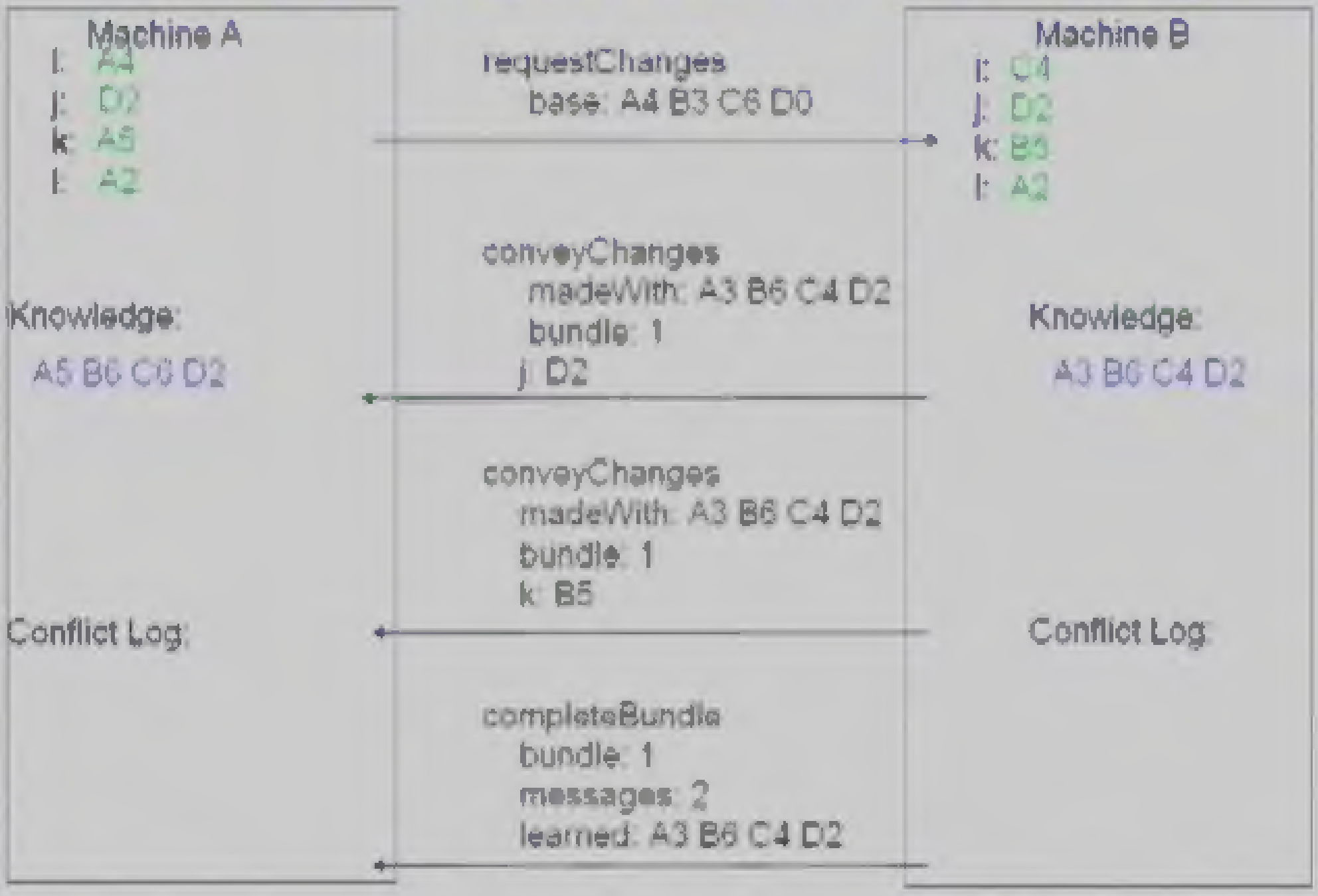
Protocol with Implicit "Made With" Knowledge



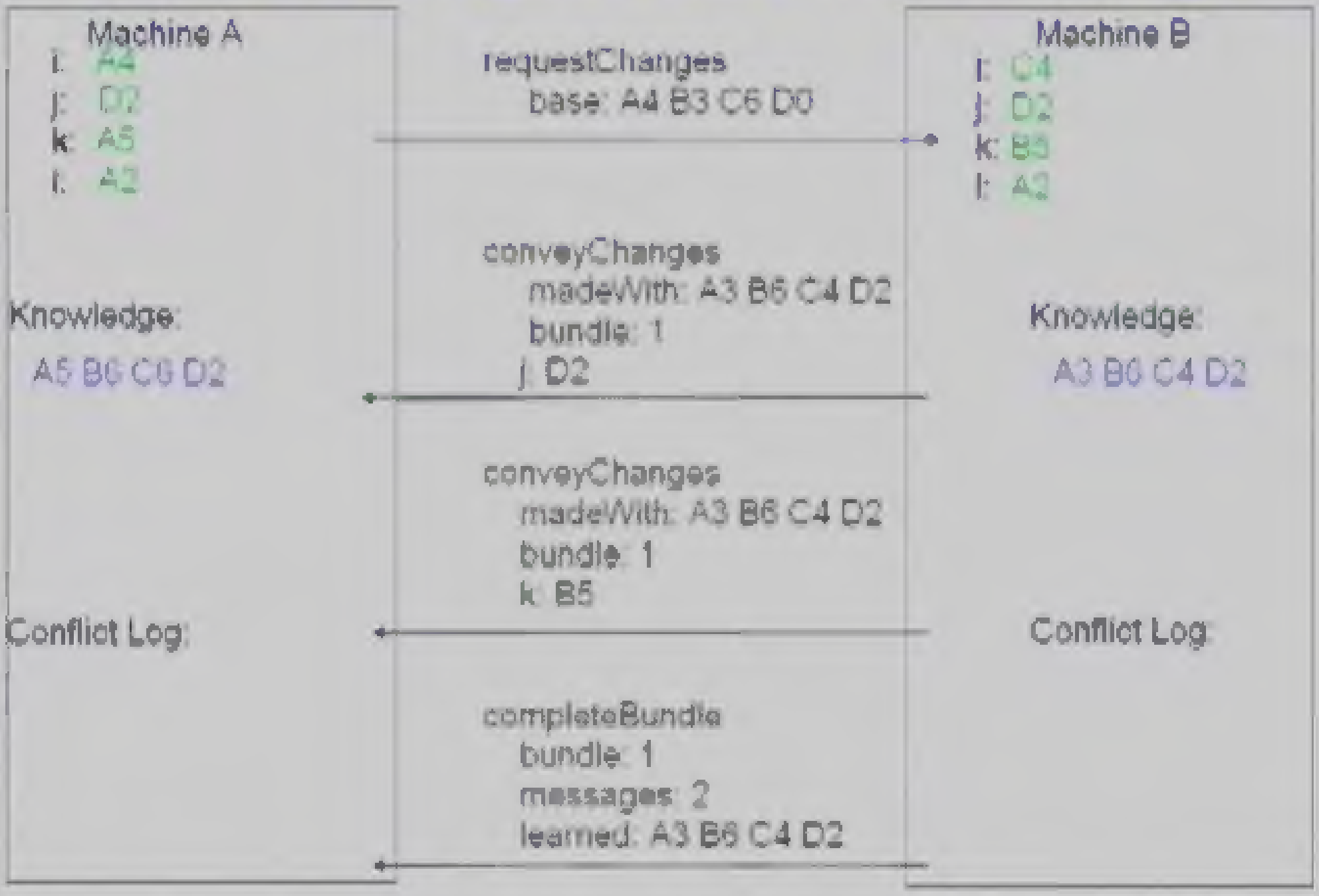
Protocol with Implicit "Made With" Knowledge



Protocol with Implicit "Made With" Knowledge



Protocol with Implicit "Made With" Knowledge



Summary

Key features of WinFS Synchronization:

- Favors availability over consistency
- Achieves robustness and scalability through pair-wise, knowledge-driven protocol
- Makes efficient use of bandwidth and storage
- Allows arbitrary communication topology
- Guarantees eventual convergence if well-connected topology
- Detects conflicts; permits manual or automatic resolution